



Inhalt

Über TaleTime

Kurze Einführung

01



Technische Anforderungen

Technologien mit denen wir arbeiten
mussten

02



Projekt-Ziele

Ziele die wir erreichen wollten

03



04

Features

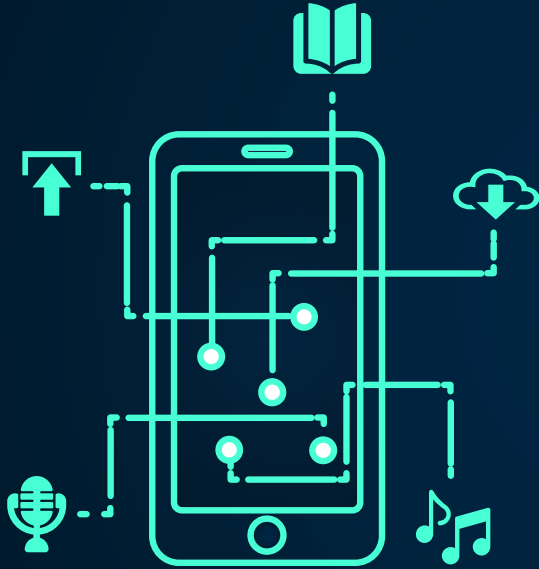
Vorstellen der wichtigsten Features und
einige technische Hintergründe



05

Live Demo

Präsentation der Live-Demo



Über TaleTime

TaleTime is an interactive audiobook app for iOS and Android based on Flutter.

Technische Anforderungen



GitHub

Projekt zu Github
hinzugefügt



Flutter

Entwicklung mit
Flutter



Firebase

Projekt mit Firebase verbunden
(Datenbank & Userverwaltung)

Projekt-Ziele



ERFAHRUNG SAMMELN

Am Ende des Projekts
sind wir mit der
Entwicklung von Apps
vertraut



FUNKTIONALITÄT

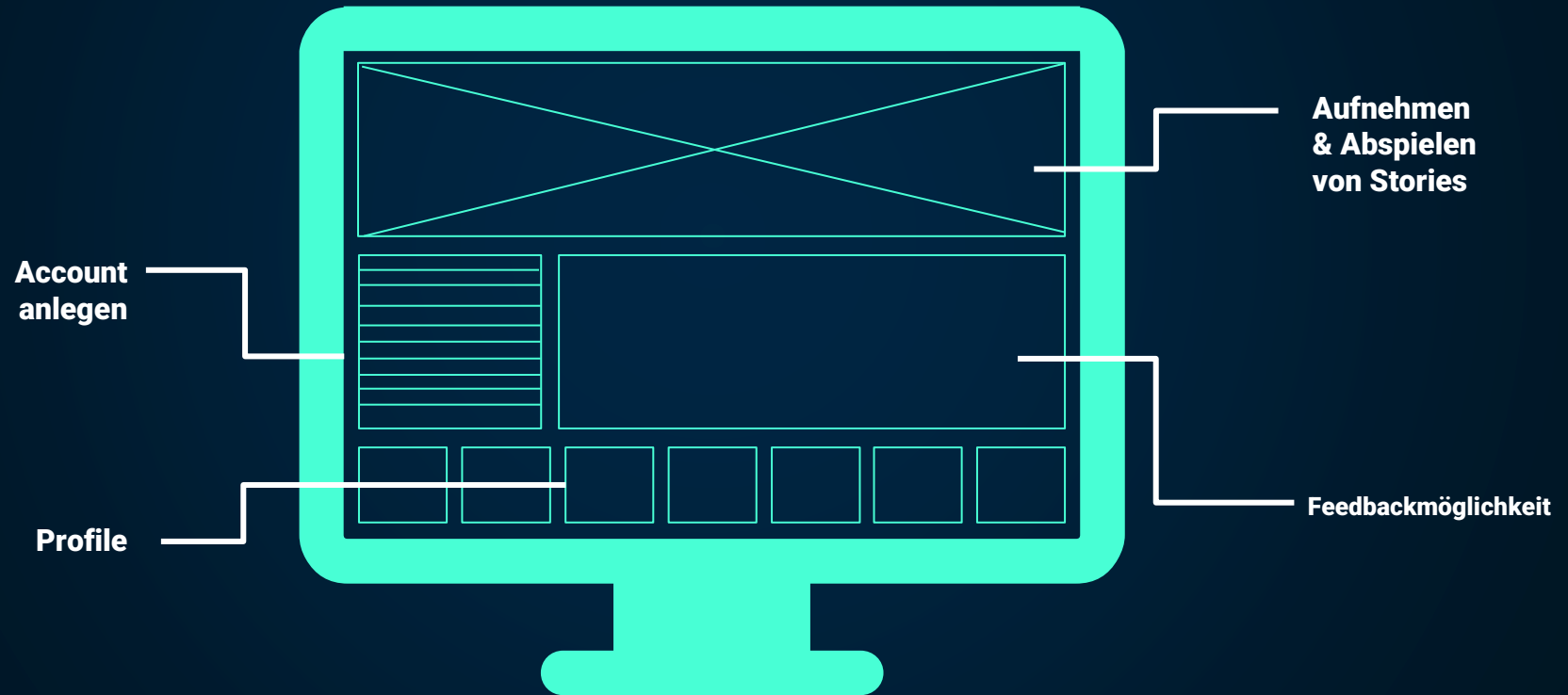
Am Ende des Projekts
sollten alle erwarteten
Funktionalitäten
implementiert sein



BENUTZERFREUNDLICHKEIT

Am Ende des Projekts
sollte die Anwendung
einfach und intuitiv zu
bedienen sein.

Features




Features: Account anlegen

Als Nutzer möchte ich einen (Mehrbenutzer) Account anlegen können

Registrierung

Erstellen Sie ein kostenloses Konto



Benutzername

Email-Adresse

Passwort


Passwort bestätigen

Registrieren

Sie haben bereits ein Konto? [Anmelden](#)

Registrierung

Erstellen Sie ein kostenloses Konto



Benutzername
testuser

Email-Adresse
test@user.de

Passwort

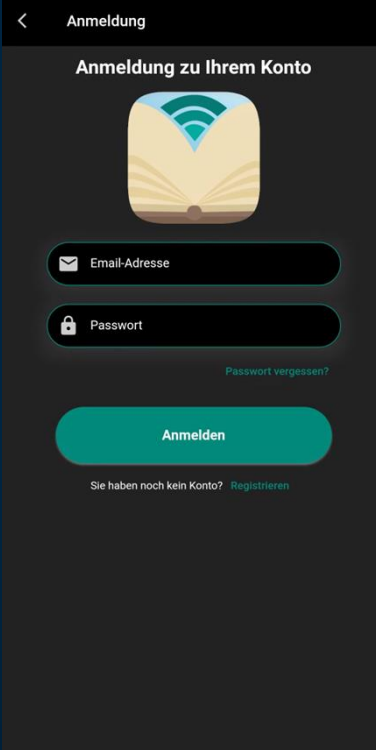
Passwort bestätigen

Registrieren

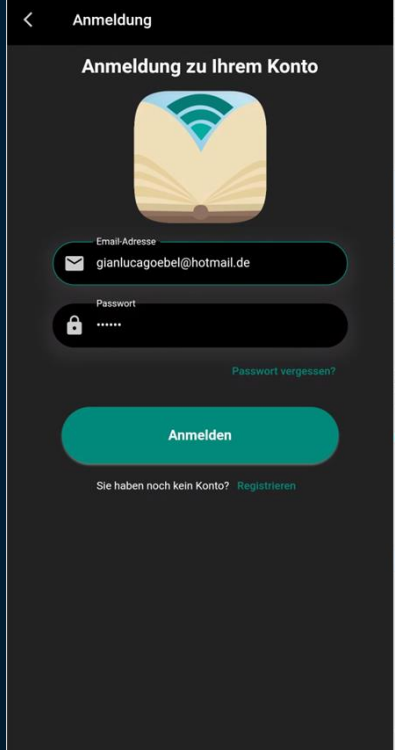
Sie haben bereits ein Konto? [Anmelden](#)

Features: Einloggen

Als Nutzer möchte ich mich bei meinem Account einloggen können



Mockup of a login screen titled "Anmeldung". The screen displays the app icon (a yellow book with a blue Wi-Fi signal) and the text "Anmeldung zu Ihrem Konto". Below the icon are two input fields: "Email-Adresse" and "Passwort". A link "Passwort vergessen?" is located below the password field. At the bottom is a large green button labeled "Anmelden". Below the button is the text "Sie haben noch kein Konto? [Registrieren](#)".



Mockup of a login screen titled "Anmeldung", identical to the previous one but with pre-filled data. The "Email-Adresse" field contains "gianlucagoebel@hotmail.de" and the "Passwort" field contains "*****". The "Anmelden" button and the "Registrieren" link remain the same.

Technischer Hintergrund: Registrieren & Login

```
/// Allows the login by entering email and password.
///
/// catches all FirebaseAuthExceptions und outputs them in the form of a s
///
/// if the login was successful then the user receives a confirmation that
Future<void> loginUsingEmailAndPassword(
  {required String email,
   required String password,
   required BuildContext context}) async {
  try {
    User? user;
    UserCredential userCredential = await auth.signInWithEmailAndPassword(
      email: email, password: password);
    user = userCredential.user;
    if (user != null) {
      final Snackbar signinSucsesful = Snackbar(
        content: Text(AppLocalizations.of(context)!.signInSucsesful),
        backgroundColor: kPrimaryColor); // Snackbar
      ScaffoldMessenger.of(context).showSnackBar(signinSucsesful);
      Navigator.of(context).pushReplacement(MaterialPageRoute(
        builder: (context) => ProfilesPage(auth.currentUser!.uid))); // M
    }
  } on FirebaseAuthException catch (e) {
    final Snackbar snackBar = ErrorUtil().showLoginError(e, context);
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }
}
```

Login

```
/// catches all FirebaseAuthExceptions and outputs them in the form of a s
///
/// if the registration was successful then the user receives a confirmati
Future<void> registerWithEmailAndPassword(
  {required String userName,
   required String email,
   required String password,
   required BuildContext context}) async {
  try {
    UserCredential userData = await auth.createUserWithEmailAndPassword(
      email: email, password: password);
    User? user = userData.user;
    user?.updateDisplayName(userName);

    Map<String, dynamic> userInfoMap = {
      "email": email,
      "password": password,
      "userName": userName,
      "UID": auth.currentUser!.uid,
    };

    if (user != null) {
      final Snackbar signupSucsesful = Snackbar(
        content: Text(AppLocalizations.of(context)!.signUpSucsesful),
        backgroundColor: kPrimaryColor); // Snackbar
      ScaffoldMessenger.of(context).showSnackBar(signupSucsesful);

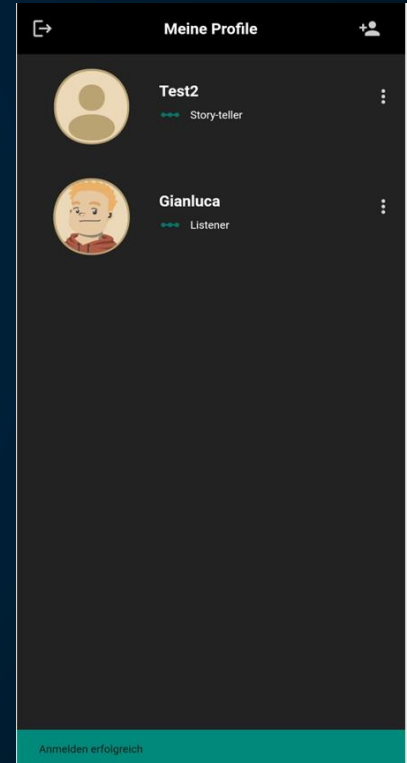
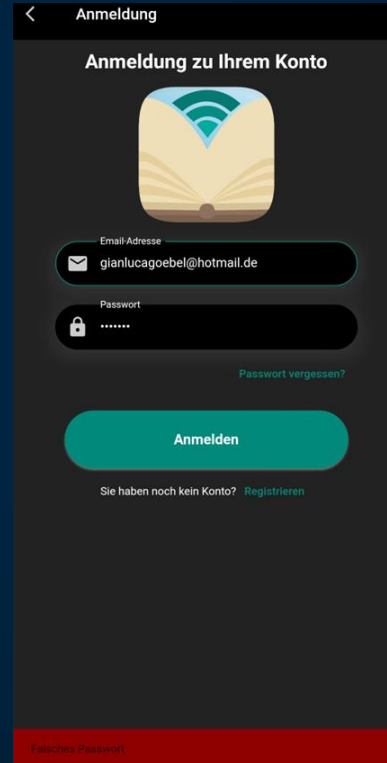
      addUserInfoToDB(auth.currentUser!.uid, userInfoMap);

      Navigator.of(context).pushReplacement(MaterialPageRoute(
        builder: (context) => ProfilesPage(auth.currentUser!.uid))); //
    }
  } on FirebaseAuthException catch (e) {
    final Snackbar snackBar = ErrorUtil().showRegisterError(e, context);
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }
}
```

Registrierung

Features: Feedbackmöglichkeit

Als Nutzer möchte ich
Feedback über falsche
oder erfolgreiche
Eingaben erhalten



Technischer Hintergrund: Feedbackmöglichkeit

```
/// Outputs the LoginPage errors in the form of a Snackbar
SnackBar showLoginError(FirebaseAuthException e, BuildContext context) {
  final SnackBar snackBar;
  if (e.code == 'user-not-found') {
    snackBar = SnackBar(
      content: Text(AppLocalizations.of(context)!.userNotFound),
      backgroundColor: kErrorColor); // SnackBar
  } else if (e.code == 'wrong-password') {
    snackBar = SnackBar(
      content: Text(AppLocalizations.of(context)!.wrongPassword),
      backgroundColor: kErrorColor); // SnackBar
  } else {
    snackBar = const SnackBar(content: Text("null"));
  }
  return snackBar;
}
```

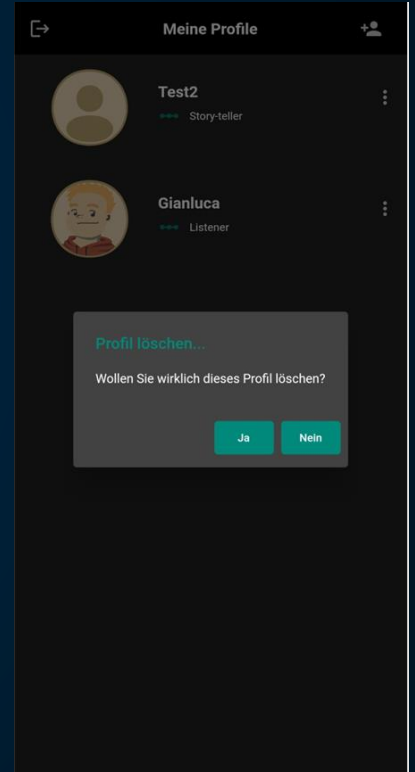
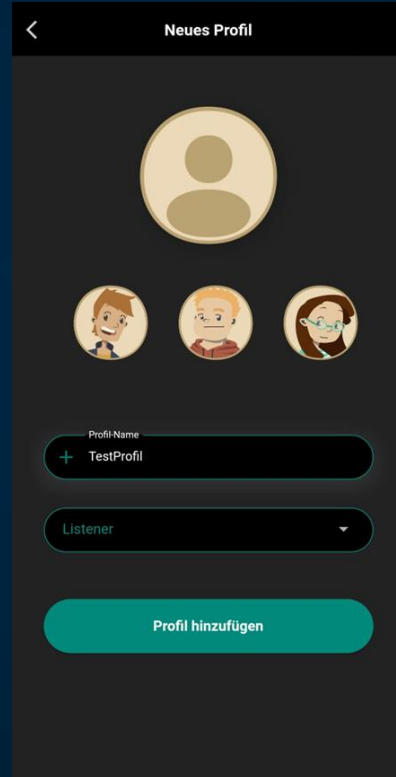
Wird bei Fehler aufgerufen

```
if (user != null) {
  final SnackBar signupSuccessful = SnackBar(
    content: Text(AppLocalizations.of(context)!.signUpSuccessful),
    backgroundColor: kPrimaryColor); // SnackBar
  ScaffoldMessenger.of(context).showSnackBar(signupSuccessful);
}
```

Wird bei erfolgreicher Anmeldung aufgerufen

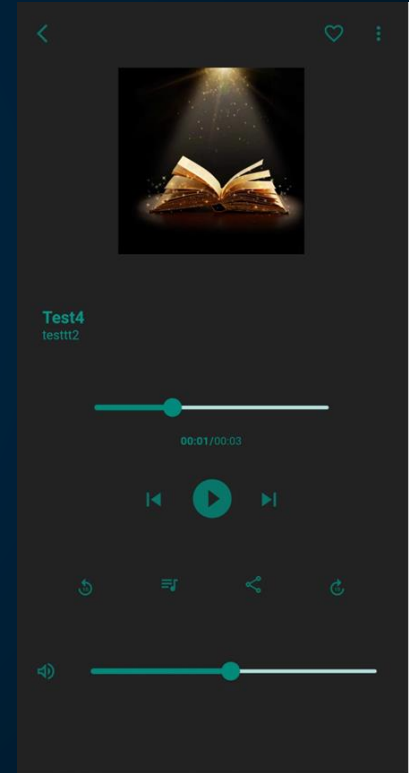
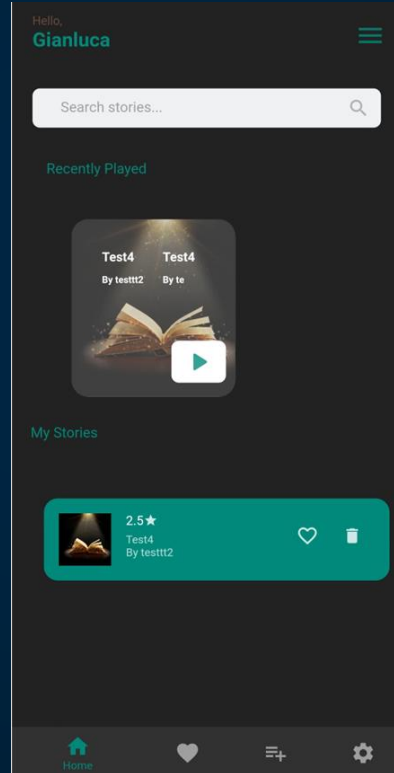
Features: Profil erstellen/bearbeiten/löschen

Als Nutzer möchte ich
ein Profil erstellen,
bearbeiten und löschen
können



Features: Geschichte abspielen

Als Hörer möchte ich
eine Geschichte
abspielen können



Technischer Hintergrund: Geschichte abspielen

```
bool isPlaying = false;

final AudioPlayer player = AudioPlayer();

double changeVoice = 0.0;
double _currentValue = 0;

Duration? duration = Duration(seconds: 0);

void initPlayer() async {
  await player.setSource(UrlSource(story["audio"]));
  duration = await player.getDuration();
}

@override
void initState() {
  super.initState();
  initPlayer();
}
```

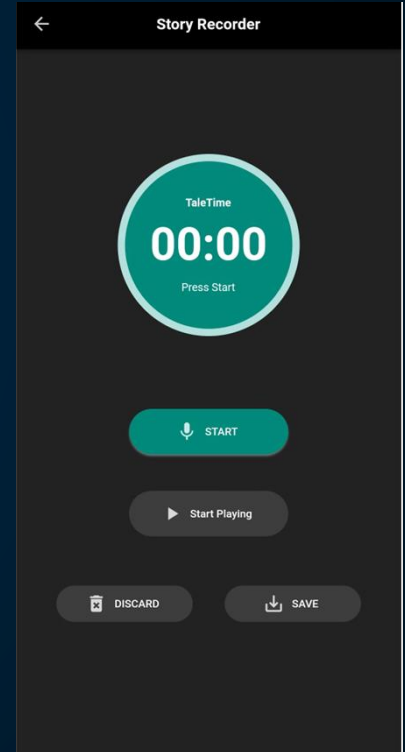
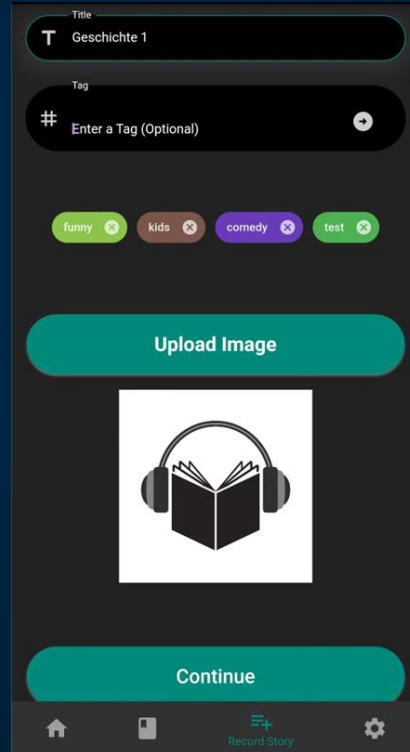
AudioPlayer initialisieren

```
onPressed: () async {
  if (isPlaying) {
    await player.pause();
    setState(() {
      isPlaying = false;
    });
  } else {
    await player.resume();
    setState(() {
      isPlaying = true;
    });
    player.onPositionChanged.listen((position) {
      setState(() {
        _currentValue = position.inSeconds.toDouble();
      });
    });
  }
  duration = await player.getDuration();
},
```

Geschichte abspielen

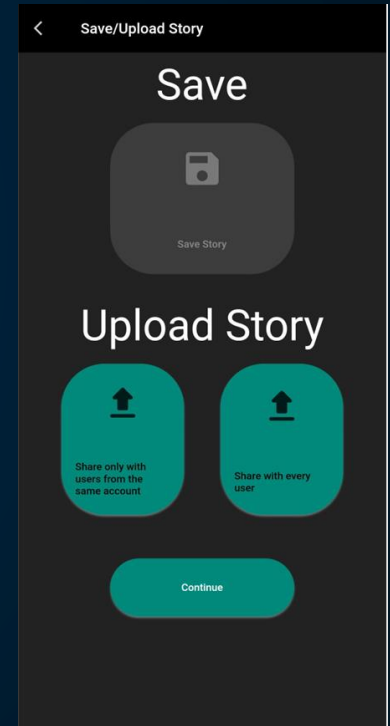
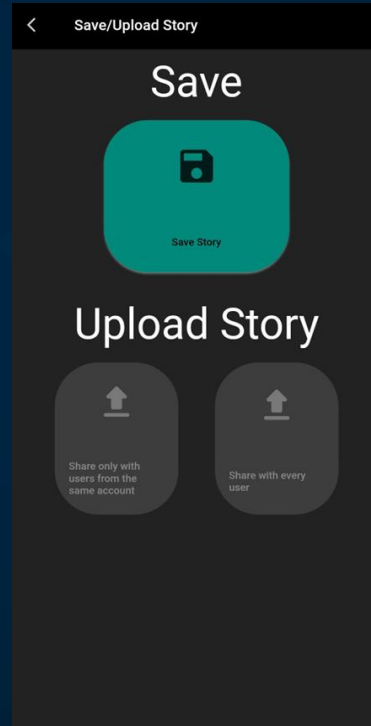
Features: Geschichte aufnehmen

Als Sprecher möchte ich eine Geschichte aufnehmen können



Features: Geschichte speichern/hochladen

Als Sprecher möchte ich eine Geschichte speichern und hochladen können



Technischer Hintergrund: Geschichte aufnehmen

```
class SoundRecorder extends FlutterSoundRecorder {
  /// FlutterSoundRecorder Object
  FlutterSoundRecorder? _audioRecorder = FlutterSoundRecorder();
  String _path = '';

  String get getPath => _path;

  bool _isRecorderInitialised = false;

  /// return true if the recorder is currently recording
  bool get isRecording => _audioRecorder!.isRecording;

  /// initializes the Sound-Recorder and asks for microphone permission if it wa
  Future initRecorder() async {
    /// asks the user for microphone permission
    final status = await Permission.microphone.request();

    if (status != PermissionStatus.granted) {
      throw RecordingPermissionException('Microphone permission not granted!');
    }

    final statusStorage = await Permission.storage.status;
    if (!statusStorage.isGranted) {
      await Permission.storage.request();
    } else {
      Directory directory = await getApplicationDocumentsDirectory();
      String filepath = directory.path +
        '/' +
        DateTime.now().microsecondsSinceEpoch.toString() +
        '.aac';

      await _audioRecorder!.openRecorder();
      _path = filepath;
      await File(_path).create();
      _isRecorderInitialised = true;

      await _audioRecorder!.setSubscriptionDuration(
        const Duration(milliseconds: 100),
      );
    }
  }
}
```

Recorder-Klasse

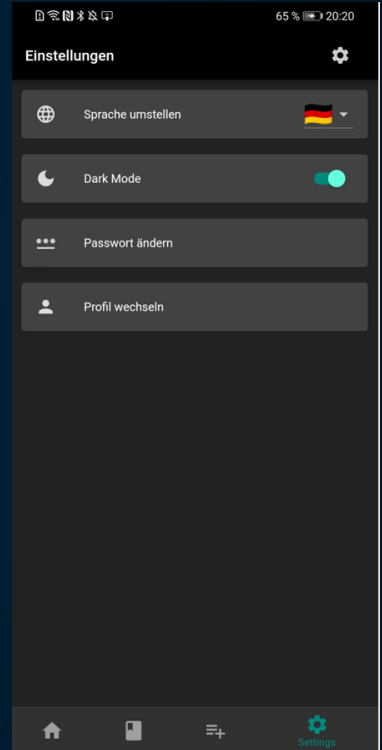
```
Future record() async {
  if (!_isRecorderInitialised) return;
  await _audioRecorder!.startRecorder(toFile: _path);
}

Future<void> stop() async {
  if (!_isRecorderInitialised) ;
  await _audioRecorder!.stopRecorder();
}
```

Aufnahme & Stop-Funktion

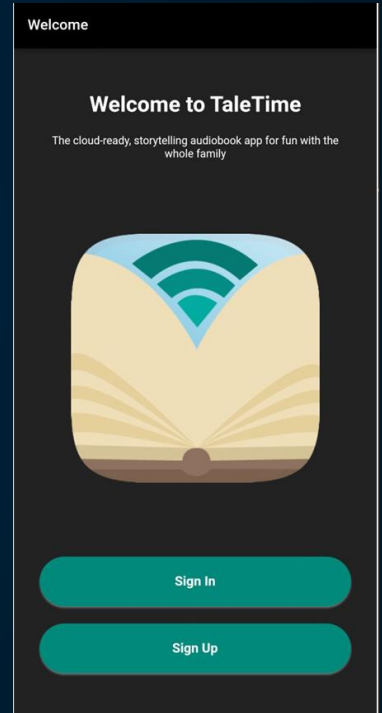
Features: Einstellungen

Als Nutzer möchte ich
die Einstellungen der
App ändern können
z.B Dark-/Lightmode
oder das Passwort



Feature: Sprache umstellen

Als Nutzer möchte ich
die Sprache der App
umstellen können



Technischer Hintergrund: Sprache umstellen

internationalization

app_ar.arb

app_de.arb

app_en.arb

l10n.dart

locale_provider.dart

```
{
  "welcomeToTaleTime": "Welcome to TaleTime",
  "@welcomeToTaleTime": {
    "description": "Willkommen bei TaleTime"
  },
  "descriptionWelcome": "The cloud-ready, storytelling audiobook app for fun with the wh",
  "@descriptionWelcome": {
    "description": "Das cloudfähige, geschichtenerzählende Vorlesetool für Spaß mit de",
  },
  "register": "Sign Up",
  "@register": {
    "description": "Registrierung"
  },
  "registerVerb": "Sign Up",
  "@registerVerb": {
    "description": "Registrieren"
  },
  "loginVerb": "Sign In",
  "@loginVerb": {
    "description": "Anmelden"
  },
  "login": "Sign In",
  "@login": {
    "description": "Anmeldung"
  },
}
```

```
1 import 'app_localizations.dart';
2
3 /// The translations for English ('en').
4 class AppLocalizationsEn extends AppLocalizations {
5   AppLocalizationsEn([String locale = 'en']) : super(locale);
6
7   @override
8   String get welcomeToTaleTime => 'Welcome to TaleTime';
9
10  @override
11  String get descriptionWelcome => 'The cloud-ready, storytelling audiobook app for fun
12
13  @override
14  String get register => 'Sign Up';
15
16  @override
17  String get registerVerb => 'Sign Up';
18
19  @override
20  String get loginVerb => 'Sign In';
21
22  @override
23  String get login => 'Sign In';
24
25  @override
```

```
mainAxisAlignment: MainAxisAlignment.spaceEvenly,
crossAxisAlignment: CrossAxisAlignment.stretch,
children: <Widget>[
```

```
  Column(
```

```
    children: <Widget>[
```

```
      Text(
```

```
        AppLocalizations.of(context)!.welcomeToTaleTime,
```

```
        style: const TextStyle(
```

```
          fontWeight: FontWeight.bold,
```

```
          fontSize: 30,
```

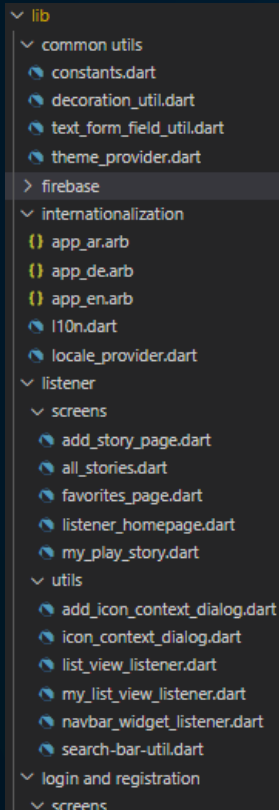
```
String get welcomeToTaleTime
```

```
package:flutter_gen/gen_l10n/app_localizations.dart
```

```
Willkommen bei TaleTime
```

```
In en, this message translates to: 'Welcome to TaleTime'
```

Technischer Hintergrund: Programm-Struktur

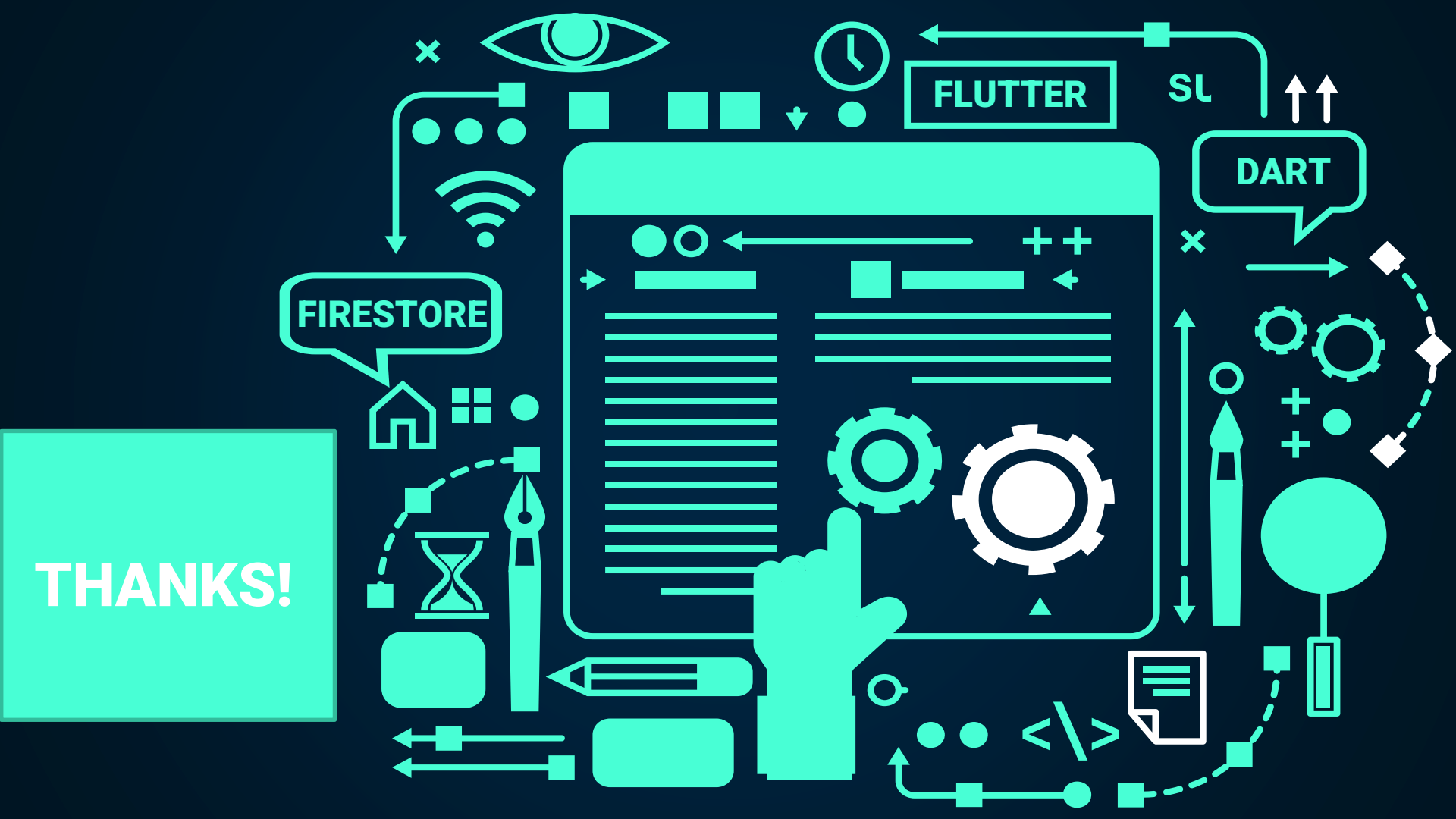


```
lib
├── common utils
│   ├── constants.dart
│   ├── decoration_util.dart
│   ├── text_form_field_util.dart
│   └── theme_provider.dart
├── firebase
├── internationalization
│   ├── app_ar.arb
│   ├── app_de.arb
│   ├── app_en.arb
│   ├── l10n.dart
│   └── locale_provider.dart
├── listener
│   └── screens
│       ├── add_story_page.dart
│       ├── all_stories.dart
│       ├── favorites_page.dart
│       ├── listener_homepage.dart
│       └── my_play_story.dart
├── utils
│   ├── add_icon_context_dialog.dart
│   ├── icon_context_dialog.dart
│   ├── list_view_listener.dart
│   ├── my_list_view_listener.dart
│   ├── navbar_widget_listener.dart
│   └── search-bar-util.dart
└── login and registration
    └── screens
```

Jedes Feature hat einen eigenen Ordner und darin:

- einen Screens-Ordner (→ UI)
- einen Utils-Order (→ Funktionen & Logik)

Live Demo



THANKS!

FLUTTER

DART

FIRESTORE

SL